VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2020 Marek Bauer

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Marek Bauer**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: ComProMiS, s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Mgr. Jiří Dvorský, Ph.D.**

Konzultant bakalářské práce: Ing. Roman Potoczný

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Tradujia elektrotechniky a informatiky 1D16 OSTRAVA

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

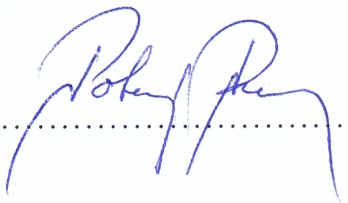
prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 20. dubna 2020 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 20. dubna 2020



Rád bych tímto poděkoval svému kolegovi Mariánovi Kluzovi, který se mnou na projektu spolupracoval a svému vedoucímu Romanu Potocznému za podmětné připomínky a konzultace.

Abstrakt

Vedení docházky je jednou ze základních povinností každého zaměstnance firmy nebo instituce. V dnešní době je již tento proces výrazně automatizován skrze čipové karty. Zadávání příchodů a odchodů, plánování dalších pracovních nebo nepracovních aktivit, schvalování těchto aktivit tvoří dohromady docházkový systém. Právě tímto se zabývá tato bakalářská práce.

Tato práce bude obsahovat popis aktuálního stavu docházkového systému ve firmě ComProMiS, s.r.o., ve které jsem absolvoval odbornou praxi, vizi nového elektronického systému, návrh struktury databáze, seznam zvolených technologií a nástrojů a na závěr popis vývoje webového rozhraní pro tento systém.

Klíčová slova: docházkový systém, aktivita, webová aplikace

Abstract

Attendance management is one of the most basic responsibilities of every comapany or institution empolyee. Nowadays, the process is significantly automated by chip cards. Inputing time of arrive and leave, planning other work related or unrelated activities, approveing those activities makes attendance system. This is the main topic of the bachelor work.

This work will contain description of actual state of attendance system in company ComProMiS, s.r.o., where i was at professional practice, vision of new electronic attendance system, design of database structure, list of chosen technologies and tools and at the end description of creation of web interface for that system.

Keywords: attendance system, activity, web application

Obsah

Seznam použitých zkratek a symbolů

ASP	–	Active Server Pages
CSS	–	Cascading Style Sheet
HTML	–	Hyper Text Markup Language
MS SQL	–	Microsoft Structured Query Language
MVC	–	Model View Controller
NFC	–	Near Field Communication

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Docházkový systém je souhrné označení činností související s plněním a evidencí docházky zaměstnanců firmy. V praxi byla tato problematika řešena několika způsoby a modernizována postupně s nástupem nových technologií. Jedním ze způsobů byla například docházková karta, která se při příchodu a odchodu proděravila přístrojem umístěným u vchodu. Dalším způsobem by mohla být například docházková kniha, kde každý den obsahuje seznam zaměstnanců spolu s jejich příchody a odchody. Tyto dva způsoby mohou být příkladem papírové evidence docházky. S příchodem nových technologií a zejména počítačů se můžeme bavit o tzv. elektronické evidenci docházky. V dnešní době spojené s technologií NFC, která slouží k bezdrátovému kontaktnímu přenosu dat. Pro přenos dat se používají karty s implementovanými čipy, které komunikují po přiložení se vstupním terminálem. Tyto systémy taktéž často obsahují desktopovou nebo webovou aplikaci pro administraci. Další variantou elektronické vedení docházky může být i mobilní zařízení. Tato možnost nabízí kromě zadávání příchodů a odchodů další funkce jako například volbu typu aktivit, plánování aktivit a další.

Úvodem bych chtěl napsat několik málo řádků o firmě, v které jsem mohl poslední dva semestry absolvovat odbornou praxi. Jedná se o firmu ComProMiS, s.r.o., která má aktuálně sídlo v Ostravě, část Kunčičky. Je to tuzemská firma s několikaletou odbornou praxí zaměřující se na tvorbou specializovaných informačních systémů a jejich údržbu. Počet zaměstnanců firmy se pohybuje okolo dvou desítek, což je potřeba brát v potaz při řešení otázky docházkového systému. Podrobnější informace o firmě budou uvedeny u představení společnosti.

Jedna z prvních kapitol bude obsahovat popis aktuálního stavu docházkového systému firmy. Jednak práva a povinnosti zaměstnanců v otázce docházky vycházející z pracovní smlouvy, ale také technologické postupy jakými je vedena jejich evidence. Mezi tyto informace například patří počet odpracovaných hodin za měsíc, pracovní doba, typy aktivit zaměstnanců, počet dnů dovolené, různé benefity vznikající z plnění docházky, způsob zápisu příchodů a odchodů a mnoho dalších informací specifických pro danou firmu.

Další kapitola bude popisem toho, jaká je vize nového elektronického docházkového systému. Na jaké platformy bude systém cílit, jaké budou funkce systému, výčet uživatelských rolí a důvod vývoje tohoto systému. Jedním z úkolů byl taktéž návrh struktury databáze. Návrhem databázového modelu včetně jeho popisu se bude věnovat jedna z následujících kapitol.

Dále bude následovat seznam použitých technologií, nástrojů a postupů použitých pro vývoj webové aplikace. K technologiím bude uveden teoretický úvod, důvod použití této technologie v projektu, popřípadě ukázka použití.

Hlavním částí práce bude popis vývoje webové aplikace, kterou jsem se zabýval většinu času mé odborné praxe. Jednotlivé funkce webové aplikace jsou chronologicky seřazeny, tak jak byly reálně přidávány. Bude popsán vývoj jednotlivých funkcí podpořený grafickým znázorněním výsledného produktu, popřípadě zdrojovým kódem. Při vývoji nastaly taktéž různé problémy, jejichž popis a řešení budou zde taktéž uvedeny.

2 Představení společnosti a pracovní zařazení

2.1 ComProMiS, s.r.o.

Společnost ComProMiS, s.r.o. se specializuje na zakázkovou tvorbu software a údržbu specializovaných informačních systémů. Má dlouhodobé zkušenosti s tvorbou informačních systémů, klient-server aplikací, intranetových aplikací, webových stránek a aplikací pro mobilní telefony.

Je partnerem společnosti Software AG a jako jedna z mála v České republice a na Slovensku umí tvořit software na operačním systému Unix (HP-UX, AIX) a na bázi databáze Adabas a vývojového prostředí Natural.

Je to firma s dnes již téměř dvacetiletou historii. Nyní se soustřeďuje převážně na vývoj v prostředí operačního systému Windows a v moderních techonologiích na bázi .NET Framework, jako jsou ASP.NET, MVC, C# a Java v databázových prostředích Microsoft SQL Server nebo Oracle. Většina jejich aplikací cílí do prostředí B2B a B2C, tedy do prostředí podpory nákupu, zpracování zakázek a výměny informací mezi centrálami a pobočkami, včetně podpory nákupu přes moderní portály. Nemalou část jejich aktivit tvoří i tvorba aplikací pro podporu servisních aktivit. [1]

2.2 Pracovní zařazení studenta

Ve firmě ComProMiS, s.r.o. jsem pracoval na pozici programátora webové aplikace. Jak již z úvodu vyplynulo, měl jsem se podílet na vývoji nového docházkového systému. Mým úkolem bylo pochopit principy vedení docházky ve firmě, získat vizi nového systému, navrhnout strukturu databáze a implementovat webové rozhraní pro nový elektronický docházkový systém. V zimním semestru jsem spolupracoval na projektu s jedním kolegou z VŠB–TUO. V letním semestru jsem již na projektu pracoval samostatně. V průběhu praxe jsme měli pravidelné konzultace s vedoucím práce. Ze začátku sloužily konzultace k získání povědomí o tom, jak by měl nový systém vypadat. V pozdější fázi, když jsem pracoval samostatně na webové aplikaci, sloužily konzultace k ověření provedené práce a domluvě na dalším postupu. Při práci na projektu jsem se setkal s technologiemi jako C#, ASP.NET, MVC, MS SQL, Azure.

3 Seznam zadaných úkolů

Tato kapitola obsahuje seznam úkolů, které jsem měl v rámci praxe zpracovat spolu s jejich analýzou. Tabulka 1 pak znázorňuje jejich reálnou časovou náročnost.

3.1 Pochopit princip vedení docházky a vytvořit vizi nového systému na základě požadavků

Začátkem práce na projektu bylo pochopení principu vedení docházky. Toho bylo docíleno konzultací se sekretariátem firmy. Výsledkem této práce měla být zpráva, která popisuje principy a postupy vedení docházky ve firmě. Tato zpráva byla následně konzultována s vedoucím projektu. Zpráva vznikla ve třech verzích. Kapitola 4 je přepisem finální verze této zprávy.

Dalším úkolem bylo na základě požadavků firmy na nový systém vytvořit vizi tohoto systému. Požadavky firmy na systém byly docela specifické a bylo zřejmé, že má firma jasnou představu o tom, jak by měl systém vypadat. Vizí nového systému se zabývá kapitola 5.

3.2 Navrhnout strukturu databáze aplikace

Dalším krokem bylo vytvoření návrhu databázového modelu aplikace. Před tvorbou modelu bylo potřeba si uvědomit, jaké jsou hlavní entity vycházející z vize nového systému. Těmto entitám byly poté přiřazeny vlastnosti, které tvoří atributy tabulek. Na základě návaznosti entit na sebe bylo pak potřeba určit jejich relační vazby. Model byl několikrát konzultován s vedoucím projektu. Hlavní body konzultací se týkali samotné struktury modelu, tabulek a relačních vazeb mezi nimi, názvů atributů a jejich datových typů. Vývojem relačního modelu a jeho popisem se zabývá kapitola 6.

3.3 Pochopit údržbu databáze v prostředí Azure Cloud

Firma se aktuálně zaměřuje na tvorbu aplikací v prostředí Azure. Tudíž jsme si měli pomocí zkušební studentské verze vyzkoušet vytvoření a správu testovací databáze. Pro získání povědomí o nové technologii jsme si prošli různé návody na internetu týkající se zejména vytvoření databáze, připojení k této databázi a provedení několika dotazů nad touto databází. V kapitole 7 se nachází popis prostředí Azure a jeho plánovaným využitím v systému. Tato kapitola taktéž obsahuje popis a důvod výběru dalších technologií použitých pro vývoj.

3.4 Naprogramovat vlastní webovou aplikaci včetně reportingu v prostředí nejnovějšího ASP.NET MVC.

Naprogramování webové aplikace bylo hlavní náplní mé práce. Před začátkem tvorby webové aplikace jsem si zvolil, jaké technologie a postupy použiji pro vývoj. Při volbě technologií a postupů jsem vycházel i z předchozí zkušenosti, kdy jsem si zkoušel vytvořit jednoduchou webovou

aplikaci v prostředí ASP .NET MVC. Mým prvotním nápadem bylo vytvořit aplikaci po vzoru kalendáře od Googlu s tím, že bych přidal zejména administrační funkce pro potvrzování pracovní doby. Nakonec jsem od tohoto nápadu upustil a zvolil jako vzor systém, který firma vyvinula v rámci jedné ze svých zakázek. Kapitola 8 popisuje pak samotný vývoj webové aplikace spolu s návrhem a řešením různých části webové aplikace.

3.5 Naprogramovat rozhraní pro export dat do personálních a mzdových systémů

Jeden z úkolů mělo být naprogramování rozhraní pro export dat sloužící pro výpočet mezd zaměstnanců. Z analýzy dosavadního docházkového systému jsem věděl, že výpočet mezd zaměstnanců probíhá na základě vytvoření excelové tabulky, kde jsou vloženy data z docházkové knihy a sečteny odpracované hodiny. Proto jsem jako jednu z funkcí webové aplikace vytvořil rozhraní, které na základě volby kalendářního roku a měsíce vytvoří podobnou tabulku. Popis této funkce se nachází pod kapitolou 8.8 v části popisující vývoj webové aplikace.

Tabulka 1: Seznam úkolů

Úkol	**Časová náročnost**
Pochopit princip vedení docházky a vytvořit vizi nového systému na základě požadavků	4 dny
Navrhnout strukturu databáze aplikace	8 dní
Pochopit údržbu databáze v prostředí Azure Cloud	4 dny
Naprogramovat vlastní webovou aplikaci včetně reportingu v prostředí nejnovějšího ASP.NET MVC.	30 dní
Naprogramovat rozhraní pro export dat do personálních a mzdových systémů	4 dny

4 Popis dosavadního systému

Cílem této kapitoly je pochopit a popsat aktuální stav, jakým je veden docházkový systém firmy ComProMiS, s.r.o., který slouží jako podklad pro výpočet mezd.

4.1 Pracovní doba

Pracovníci mají pružnou pracovní dobu. Základní pracovní doba je od 9:00 do 14:00 a volitelná pracovní doba je od 6:00 do 9:00 a od 14:00 do 18:00. Zaměstnanci musí plnit základní pracovní dobu. Pokud není splněna, například pozdní příchod, je zaměstnanec povinen čerpat půl dne dovolené. Zbylé hodiny si mohou rozdělit v kalendářním měsíci tak, aby odpracovali stanovenou pracovní dobu za kalendářní měsíc. (7,5h × počet pracovních dnů v měsíci) Čas příchodu a odchodu se zaokrouhluje po 15 minutách následovně:

- **Příchod** se zaokrouhluje nahoru (například pokud zaměstnanec přijde 8:01, jeho pracovní doba začíná v 8:15).
- **Odchod** se zaokrouhluje dolů (například pokud zaměstnanec odejde 16:14, jeho pracovní doba končí v 16:00).

Pokud je pracovní doba alespoň 5h denně, pracovníci mají nárok na 0,5h přestávku, která se nepočítá do pracovní doby. Zároveň jim tímto dle vnitřní směrnice vzniká nárok na stravenku.

4.2 Docházka

V současnosti se docházka zaznamenává do knihy příchodů a odchodů. Za zápis do knihy jsou zodpovědní samotní pracovníci. Na konci měsíce sekretářka sečte odpracované hodiny jednotlivých zaměstnanců a vypočtené údaje zapíše do tabulky v excelu. Soupis obsahuje pro každého zaměstnance součet řádně odpracovaných hodin, hodin navíc nad rámec 7,5h pracovní doby a chybějící hodiny do 7,5 hodinové směny, přičemž v součtu musí být splněn povinný počet pracovních hodin za měsíc.

4.3 Překážky v práci na straně zaměstnance

Od běžné docházky se řeší překážky v práci na straně zaměstnance, mezi které patří například návštěva lékaře, odběry krve, ostatní důvody.

- **Návštěva lékaře** – Pokud se jedná o celodenní návštěvu lékaře, počítá se to jako jeden odpracovaný den, jinak se absence v základní pracovní době počítá jako odpracované hodiny.
- **Darování krve** – Při darování krve má zaměstnanec nárok na 24h uvolnění z práce, pokud to zasahuje do pracovní směny, tak se tyto hodiny taktéž započítávají jako odpracované.

- **Ostatní důvody** – Ostatní důvody, pokud to umožní zákon se řeší obdobně jako návštěva u lékaře.

U každého z těchto uvolnění je potřeba prokázat pravdivost dokladem.

4.4 Pracovní cesty, homeoffice, státní svátek

- **Pracovní cesta** – U pracovních cest je základní pracovní doba od 8 do 16 hodin. Pokud přijde zaměstnanec mimo tyto hodiny do práce, tak se mu hodiny připočítávají jako hodiny navíc.
- **Homeoffice** – Zaměstnanci mají dle vnitřní směrnice možnost pracovat z domova.
- **Státní svátek** – Pokud připadne státní svátek na pracovní den.

V současnosti si pracovníci pracovní cesty a dovolené plánují do webového kalendáře. Kalendář slouží sekretářce pro přehled aktivit zaměstnanců a k výpočtu jejich pracovní doby. Pracovní cesty, homeoffice i státní svátky se počítají jako odpracovaný den (7,5h).

4.5 Náhradní volno

Pokud pracovníci mimořádně pracují přesčas i o víkendu nebo svátcích, mohou si následně vybrat náhradní volno i v pracovních dnech maximálně do tří měsíců, jinak se jim odpracované hodiny proplácejí.

4.6 Nemoc, dovolená

- **Nemoc** – Počátek a ukončení nemoci se eviduje pomocí potvrzení o pracovní neschopnosti.
- **Dovolená** – Každý zaměstnanec má nárok na 25 dnů dovolené.

Pokud v měsíci čerpá dovolenou nebo nemoc, tak se povinný fond pracovní doby za daný měsíc sníží o počet dnů čerpaní dovolené a nemoci.

5 Vize nového systému

Cílem projektu je vytvořit elektronický docházkový systém sloužící k vedení a evidenci docházky zaměstnanců. Pracovní název tohoto systému je ODDO (název vzniklý spojením slov „od“ a „do“). Celá vize projektu je založena na mobilní a webové aplikaci.

Mobilní aplikace bude sloužit výhradně pro zadávání příchodů a odchodů zaměstnanců. Aplikace bude mít možnost zjistit polohu zaměstnance a na základě jeho výskytu v blízkosti firmy umožnit počátek, ukončení nebo přerušení pracovní doby. Dále bude aplikace schopna zobrazit detail aktivit a zajistit plánování aktivit jako je dovolená, pracovní cesta, návštěva lékaře, homeoffice a podobně.

Webová aplikace bude stejně jako mobilní aplikace sloužit k přidávání aktivit **zaměstnanců**, ale také k administračním úkonům. Zejména schvalování, popřípadě editace aktivit pověřenou osobou (**schvalovatel**). Schvalovatel bude mít taktéž přístup k rozhraní sloužícímu k výpočtu mezd. Dále funkce **datového manažera**, který bude moci měnit data v databázi jako je kalendář, pracovní plán a podobně viz kapitola 6, kde je popsán datový model. Poslední rolí bude **administrátor**, který bude mít přístup ke všem funkcím, které informační systém nabízí.

Webová aplikace bude umístěna na serveru a tudíž k ní bude možný přístup z jakéhokoli zařízení, v kterém lze otevřít prohlížeč[1]. Pro přístup do systému bude mít každý uživatel vytvořený uživatelský účet pomocí, kterého se bude přihlašovat.

Důvodem vývoje tohoto informačního systému je bezesporu zjednodušení celého procesu týkající se docházky. Jednak pro samotné zaměstnance, ale také pro sekretariát firmy zodpovědný za dohled nad plněním pracovní doby a výpočtu mezd. Aktuální stav docházky vedený knihou příchodů a odchodů je v počtu zaměstnanců firmy na hraně únosnosti. Pokud by firma měla výrazným způsobem rozšiřovat svoje řady, elektronická evidence docházky by byla více než žádoucí.

Vývoj mobilní aplikace měl mít na starost můj kolega ze školy, se kterým jsme první semestr pracovali na vývoji systému společně. Bohužel vlivem okolností již druhý semestr na praxi nenastoupil, tudíž bylo potřeba vyřešit vzniklý problém. Řešení bylo nakonec relativně jednoduché. V druhém semestru byl již projekt ve fázi, kdy jsme měli hotovou společnou datovou část. Právě jsme začínali pracovat každý již na vývoji své vlastní aplikace. Pro projekt tedy odchod kolegy znamená, že se bude výsledný systém skládat prozatím pouze z webového rozhraní, kde bude přidávání aktivit řešeno jednorázově. Tedy přidávání aktivit, které mají již ukončený nebo naplánovaný začátek a konec bez ověření polohy zaměstnance.

[1] Aktuálně je v plánu pouze verze pro desktopové zařízení.

6 Návrh struktury databáze

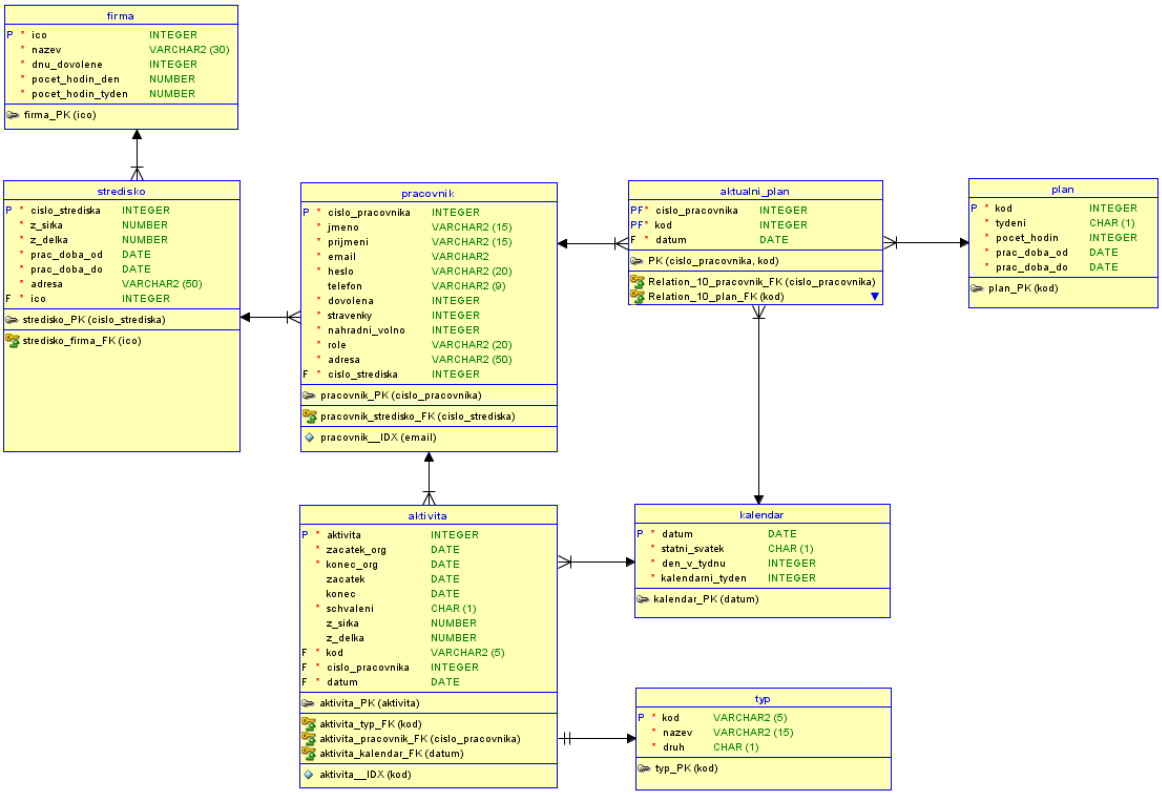
Jak již bylo řečeno databázový model byl dalším krokem po úvodním seznámení s principy vedení docházky ve firmě a získání přehledu o tom, jak by měl nový systém vypadat. První návrh modelu, který byl konzultován, se skládal pouze z několika tabulek. Po první konzultaci bylo potřeba vyřešit dva nedostatky, které model měl.

První byl ten, že systém jako takový by nebylo možné použít pro více různých firem s odlišnými pracovními podmínkami. Tudíž bylo potřeba přidat tabulky, které by tyto údaje obsahovaly.

Dalším nedostatkem byla absence pracovních plánů. Tyto plány by měly sloužit k určení pracovní doby zaměstnance a počtu hodin, které by měl plnit. Příkladem je rozdílný pracovní plán pro zaměstnance, který přišel do práce na jednu z poboček firmy a plánem pro zaměstnance, který jel na pracovní cestu. Tyto plány mají stejnou určenou délku pracovní doby, nicméně se liší její začátek a konec.

Po dalších dvou konzultacích jsme dospěli k modelu, který je možné vidět na obrázku 1.

6.1 Model



Obrázek 1: Relační model

Důležitá tabulka, které je potřeba si všimnout, je tabulka Aktivita. Ta vždy náleží jednomu pracovníkovi, má určené datum a typ. Obsahuje také dvojici údajů o začátku a konci aktivity. První dvojice značí dobu aktivity, kterou zadal pracovník. Druhá dvojice je volitelná a vyplňuje ji schvalovatel v případě, že původní doba neodpovídá realitě. Tabulka obsahuje také údaje o poloze ve formátu zeměpisné šířky a délky. Tyto údaje jsou zde uvedeny pro mobilní aplikaci, která by s nimi měla pracovat.

Tabulky Firma a Středisko jsou zde z důvodu přenositelnosti systému mezi různými firmami. Kromě počtu odpracovaných hodin za den a týden jsou zde údaje jako počet dnů dovolené nebo údaje o poloze. Údaje o poloze znovu slouží k ověření, jestli se zaměstnanec při požadavku na přidání pracovní aktivity nachází v blízkosti pobočky firmy.

Další věcí stojící za zmínku jsou tabulky týkající se plánů. Každý pracovník může mít více plánů, které jsou buď denní nebo týdenní. Plány mají určený počet hodin, počátek a konec pracovní doby, jejichž plnění se kontroluje na základě zvoleného plánu.

Kalendář je tabulka, jejíž záznamy reprezentují jeden kalendářní den. Den se samozřejmě skládá z data, z toho, jestli daný den připadá na státní svátek, údaje o tom, o jaký den v týdnu se jedná a informaci o kalendářním týdnu.

Tabulka typ obsahuje seznam různých typů aktivit. Typ je určen kódem a názvem. Atribut druh značí, do jaké skupiny typ spadá. Typy se dělí do následujících skupin.

- Do **produktivních** aktivit se řadí aktivity typu práce vykonávaná přímo ve firmě, služební cesty, homeoffice.
- **Neproduktivní** aktivity jsou návštěva lékaře, darování krve, školení a ostatní důvody.
- Mezi **absence** patří dovolená, náhradní volno a nemoc.
- **Pohotovost** je typ aktivity, kdy pracovník nemusí být přítomen přímo ve firmě, ale je připraven v určitém čase řešit vzniklý problém.

Poté co se navrhnutý model začal implementovat, bylo potřeba přidat nový primární klíč pro každou tabulku. Konkrétně atribut s názvem Id ve formátu textového řetězce. Toto vychází ze zděděné třídy **EntityData**, která se nachází v knihovně Azure Mobile. Výhodou použití primárního klíče v podobě textového řetězce je, že tento klíč je možné generovat vždy jako jedinečný. V praxi se tento typ upřednostňuje před klasickým celočíselným datovým typem, kde je riziko, že při případném sloučení databázi by mohlo docházet ke konfliktům. Původní primární klíče byly zachovány pro jejich informační hodnotu. Například původní klíč značící číslo pracovníka se využívá i při výběru zaměstnance ze seznamu.

7 Použité technologie

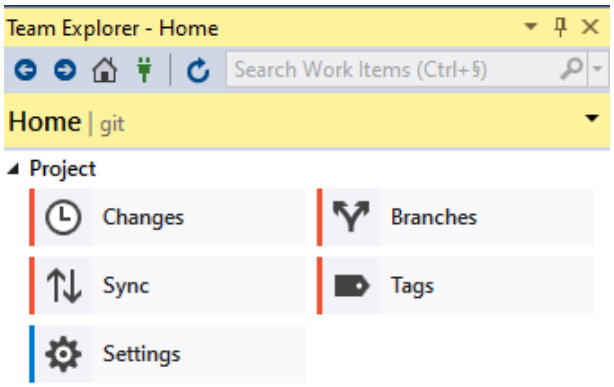
7.1 Azure

Windows Azure Platform [2] je cloudová platforma společnosti Microsoft. Využívá se k vytváření, hostování a škálování webových aplikací prostřednictvím datových center Microsoftu. Windows Azure má tři hlavní částí:

- Výpočetní část – zabývá se chodem a fungováním aplikací
- Úložiště – zajišťuje online uložení aplikací a různých dat
- Strukturní část – zajišťuje rychlé a spolehlivé propojení s internetem a datacentry Microsoftu, strukturní část zahrnuje jak výpočetní část, tak i úložiště [3]

Jak již bylo řečeno, důvodem použití této platformy byla její preference ze strany firmy, která při vývoji vlastních aplikací aktuálně cílí na její použití. Platforma Windows Azure by měla v docházkovém systému vystupovat hned v několika rolích.

První role je využití Azure DevOps. To je soubor služeb pro efektivní vývoj aplikací. Služba, která byla využita v našem projektu se jmenuje Azure Repos. Tato služba slouží k privátnímu hostování Git úložišť. Služba obsahuje nástroje, které zjednodušují operace nad sdíleným projektem. Výhodou je i podpora této služby Visual Studiem, které nabízí několik nástrojů pro práci se sdíleným repozitářem (viz Obrázek 2).

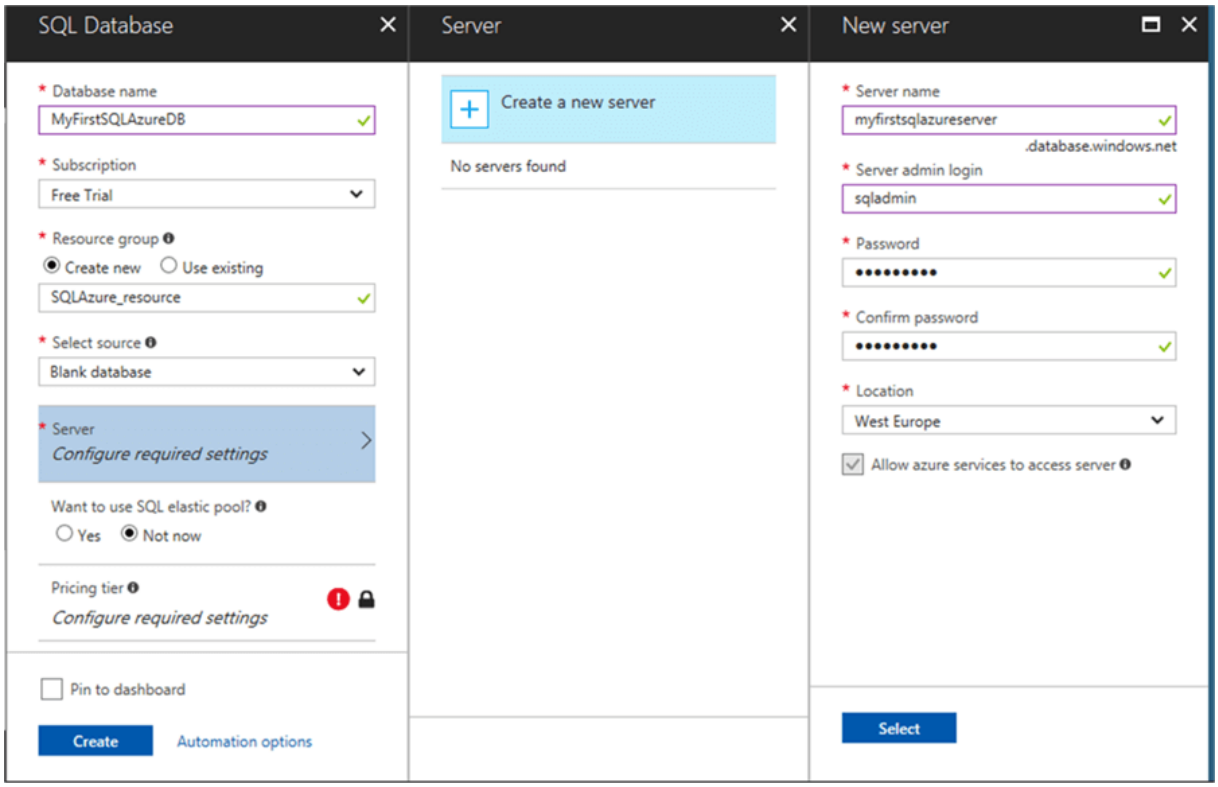


Obrázek 2: Nástroje Visual Studia

Další rolí Azure v našem projektu je využití Azure účtů při přihlašování uživatelů do systému. Při vytváření ASP .NET MVC aplikace lze zvolit z několika možností pro autentizaci uživatelů. Jednou z nich je přihlašování pomocí pracovních účtů. Mezi ně patří i účty Microsoft Azure, které zaměstnanci firmy vlastní.

Finální verze webové aplikace by taktéž měla běžet na serveru od Azure, spolu s cloudovým úložištěm pro databázi. Vytvoření testovací databáze bylo jednou z činností při seznámení se

s touto technologií. Na obrázku 3 lze vidět, jak vypadá proces vytvoření jednoduché databáze. Při vytváření lze kromě jiného zvolit do jaké skupiny zdrojů bude databáze patřit. Této skupině lze pak nastavovat například různá práva společná pro všechny prvky ve skupině. Při vytváření serveru lze taktéž zvolit preferované umístění serveru, které vyhovuje lokaci použité aplikace. Po vytvoření samotné databáze je možnost nakonfigurovat firewall, který povolí přístup k databázi pouze IP adresám, které spadají do zvoleného rozsahu.



Obrázek 3: Vytvoření databáze [4]

7.2 MS SQL

Jedná se o vyspělý databázový systém, který byl vyvinut společností Microsoft v roce 1992 ve verzi Microsoft SQL Server [5] verze 4.2. pro platformu Windows. [6]

Vývojovým nástrojem, který byl zvolen pro vytvoření databázového modelu je SQL Developer Data Modeler od Oraclu, který nabízí několik nástrojů pro tvorbu logického a relačního modelu databáze. Důvodem výběru tohoto nástroje byla dřívější pozitivní zkušenost s tímto nástrojem.

7.3 ASP .NET Framework MVC

ASP.NET MVC [7] je webový aplikační framework, který implementuje vzor Model-View-Controller (MVC). Na základě ASP.NET umožňuje vývojářům vytvářet webové aplikace jako

složení tří komponent: modelu, pohledu a řadiče. Model představuje stav určitého aspektu žádosti. Model často mapuje databázové tabulky se záznamy v tabulce zastupované podle stavu žádosti. Řadič se zabývá interakcí a aktualizací modelu, aby odrážel změny ve stavu aplikace a potom předává informace do pohledu. Pohled přijímá nezbytné informace od regulátoru a vykresluje změněné uživatelské rozhraní. [8]

Před začátkem tvorby webové aplikace jsem měl možnost výběru pomocí jaké technologie budu webovou aplikaci vytvářet. Na výběr jsem měl mezi ASP. NET Framework MVC a ASP .NET Core MVC. Výhodou verze Core je, že se jedná o novější framework, který poskytuje moderní přístup pro vytváření webových aplikací. Jeho nevýhodou je menší podpora oproti verzi Framework, a to jak ze strany dostupných materiálů na internetu, tak ze strany pracovníků firmy, kteří mají zkušenost s verzí Framework. Toto bylo hlavním důvodem, proč jsem nakonec zvolil verzi Framework pro vývoj mé webové aplikace. Tato volba se později ukázala jako dobrá při řešení různých technických otázek.

Samotná tvorba souborů v ASP .NET obsahující zdrojový kód stránky je odlišná od klasického přístupu. V tomto případě lze kombinovat programovou logiku a psaní HTML kódu. Tento způsob zajišťuje tzv. Razor.

7.3.1 Razor

Razor je syntaxe značek pro vkládání do webových stránek kód založený na serveru. Syntaxe Razor se skládá ze značek Razor, C# a HTML. Soubory obsahující Razor mají obecně příponu .cshtml. [9] Ukázku tohoto kódu lze vidět ve výpisu 1.

```
@for (var i = 0; i < people.Length; i++)
{
    var person = people[i];
    <p>Name: @person.Name</p>
}
```

Výpis 1: Ukázka .cshtml kódu

Razor používá pro přechod mezi C# kódem a HTML symbol „@“. Pokud v bloku označeném tímto symbolem chceme psát HTML element, Razor automaticky rozpozná tento formát a interpretuje jej jako HTML kód.

7.3.2 Identity

Identity [10] je podpora pro vytváření a správu uživatelských účtu. Je to šablona, která výrazným způsobem zjednodušuje práci s uživatelskými účty a jejich rolemi v systému. Její použití se volí při vytváření webové aplikace typu MVC.

Možnosti přihlašování jsou:

- Bez přihlašování
- Osobní účet
- Školní nebo pracovní účet
- Windows účet

7.4 Entity Framework

Pro objektově relační vrstvu a přístup k databázi je použit Entity Framework [11], který výrazně zjednodušuje operace týkající se připojení k databázi, selektování dat, ukládání, editaci a mazání záznamů. Entity Framework byl součástí .NET, od verze 6 je od tohoto rámce oddělen. [12]

7.4.1 CodeFirst

Při práci na projektu jsem využíval tzv. „CodeFirst“ přístup, který vychází z toho, že nejdříve programátor napíše kód ve formě modelu a na základě toho se tvoří databázová struktura. Následuje jednoduchý postup, který byl použit při využívání tohoto přístupu.

1. Vytvoření modelu, kde třídy reprezentují tabulky databáze.
2. Deklarace těchto tříd v souboru, který reprezentuje databázový kontext.
3. Vytvoření migrace pomocí příkazu „add-migration“ v příkazovém řádku Package Manager konzole. Tímto příkazem se vytvoří soubor, který obsahuje všechny změny v modelu, které se mají aplikovat na databázi.
4. Provedení změn se provádí pomocí příkazu „update-database“.

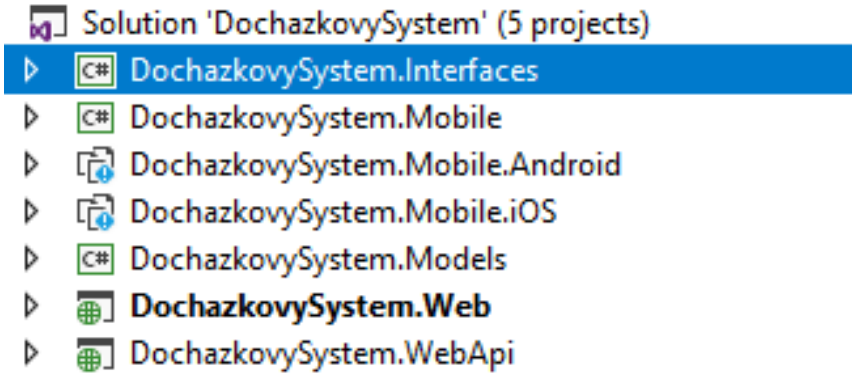
7.5 C#

V neposlední řadě programovací jazyk, ve kterém byl psán kód aplikace. C# [13] je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework. C# lze využít k tvorbě databázových programů, webových aplikací a stránek, webových služeb, formulářových aplikací ve Windows, softwaru pro mobilní zařízení (PDA a mobilní telefony) atd. [14]

8 Webová aplikace

8.1 Struktura projektu

Logicky dalším krokem po seznámení se s požadovaným systémem, technologiemi a vytvořením modelu databáze, bylo vytvoření projektu. Pro určení struktury projektu sloužila, po konzultaci s dvěma pracovníky firmy, ověřená šablona, kterou firma použila v jednom ze svých projektů. Tento projekt se taktéž skládal z webové a mobilní aplikace, tudíž struktura mohla být použita prakticky identicky. Na obrázku 4 lze vidět složení struktury.



Obrázek 4: Struktura projektu

Struktura by se dala rozdělit do tří částí:

1. Společný model
2. Webová aplikace
3. Mobilní aplikace a back-end

Společný model obsahuje knihovnu tříd, které slouží jako vzory a knihovnu tříd, které tyto vzory implementují. Tyto třídy slouží pro mapování objektů na tabulky databáze a jsou společné pro webovou a mobilní aplikaci. Výpis 2 obsahuje ukázku třídy reprezentující tabulku Kalendář. Je vidět, že třída implementuje rozhraní **IKalendar** a dědí z třídy **EntityData**, která třídě při migraci přidává atribut Id pro vytvoření unikátního primárního klíče. Pro úspěšné vytvoření databáze a propojení s modely je potřeba vytvořit soubor, který reprezentuje tzv. databázový kontext. Tento soubor obsahuje kolekci entit, které mohou být použity pro práci s databází. Přidání třídy do této kolekce se provádí pomocí funkce **Set**. Mimo jiné obsahuje tento soubor různé nastavení týkající se databáze. V jejím konstruktoru se definuje tzv. **Connection String**, který obsahuje údaje potřebné pro připojení k databázi[2].

[2]V průběhu vývoje webové aplikace jsem pracoval pouze s lokální databází.

Z počátku nastaly určité komplikace se sdíleným modelem. Po přidání třídy **EntityData**, nebylo možné provádět migrace ze strany webové aplikace. Problém bylo potřeba vyřešit přidáním konfigurace právě do souboru s databázovým kontextem. Další skutečností, která vznikla z třídy **EntityData**, byla potřeba volat funkci pro vytvoření nového Id při vytváření nového záznamu ve webové aplikaci.

```
namespace DochazkovySystem.Models
{
    public class Kalendar : EntityData, IKalendar
    {
        public DateTime Datum { get; set; }
        public bool StatniSvatek { get; set; }
        public int DenVTydnu { get; set; }
        public int KalendarniTyden { get; set; }
    }
}
```

Výpis 2: Třída Kalendář

Webová aplikace se pak skládá z jednoho projektu, který má vlastní strukturu vytvořenou na základě vzoru MVC. Základní struktura se dělí do tří částí. Vlastních modelů, kontrolerů a views. Modely webové aplikace obsahují pomocné třídy pro reprezentaci dat, které se posílají mezi kontrolerem a tím, co se má zobrazit uživateli (view).

Mobilní aplikace je o něco složitější. Při vývoji mobilních aplikací pomocí technologie Xamarin je potřeba vytvořit zvlášť projekt pro platformy, na které aplikace cílí. V tomto případě Andoid a iOS. Dále obsahuje projekt WebApi sloužící jako back-end spravující modely systému.

Pro práci ve skupině byl vytvořen společný repositář, který byl následně propojen s tímto projektem pomocí služby Azure Repos. Změny v projektu se pak projevovaly pomocí comitů a synchronizace projektů za pomocí nástrojů ve Visual Studiu. Ze začátku nastaly i nepatrné problémy v této oblasti. Problém nastal po změně určitých tříd a pokusu o synchronizaci projektu, která neproběhla úspěšně. Řešení tohoto problému trvalo skoro jeden pracovní den. Nakonec pomohlo naklonování celého projektu z repositáře a znovu připojení k repositáři.

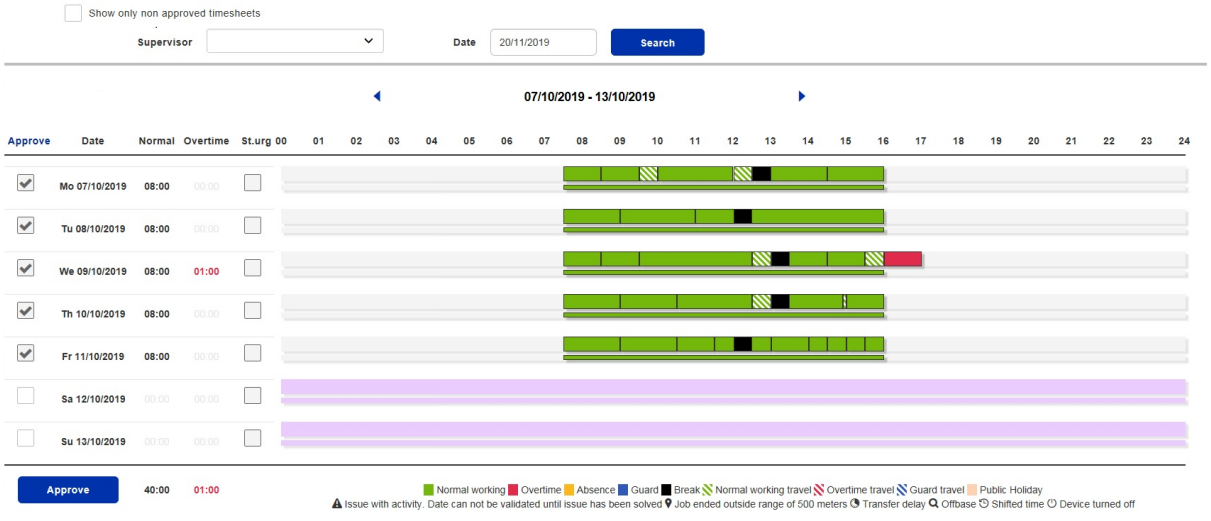
8.2 Generování kalendáře

První funkcí po vytvoření projektu a vyřešení problému se sdíleným modelem byla funkce pro automatické generování záznamů v tabulce Kalendář. Záznamy této tabulky reprezentují jeden kalendářní den. Bylo by velice neefektivní, kdyby datový manažer aplikace byl nucen všechny dny spolu s jejich vlastnostmi zadávat do databáze ručně. Proto byla vytvořena funkce, která tyto záznamy tvoří automaticky. Rozhraní pro tuto funkci se skládá pouze z jednoduchého formuláře,

který obsahuje pole pro zadání počátečního a koncového data. Ve výpisu 5 (viz příloha B) je pak možné vidět, že funkce využívá knihovnu **Nager.Date**, která na základě kódu země dokáže určit, je-li daný den státní svátek nebo ne. Pro určení kalendářního týdne se využívá třída **Calendar**. Samozřejmě se před přidáním nového záznamu testuje, jestli není záznam pro dané datum již vytvořen.

8.3 Vizualizace kalendáře

Po naplnění tabulky kalendáře bylo možné začít pracovat na vizualizaci tohoto kalendáře a přidávání aktivit. Před tímto úkonem bylo vhodné mít představu o tom, jak by tato důležitá část projektu měla vypadat. Proces vytváření skic systému trval asi dva pracovní dny. Prvotní návrh projektu se skládal z měsíčního přehledu dnů, kdy po výběru dne se zobrazil souhrn aktivit za daný den spolu s formulářem pro přidání nové aktivity. Po konzultaci těchto návrhů s vedoucím projektu vznikl požadavek na změnu tohoto návrhu. Výsledný systém se měl skládat z týdenního kalendáře namísto měsíčního. Tento týden měl zároveň zobrazovat aktivity za jeden pracovní den zaměstnance na časové ose. Pro vzor nakonec posloužil systém, na kterém firma v minulosti pracovala. Na obrázku 5 je screenshot z tohoto systému.



Obrázek 5: Skica

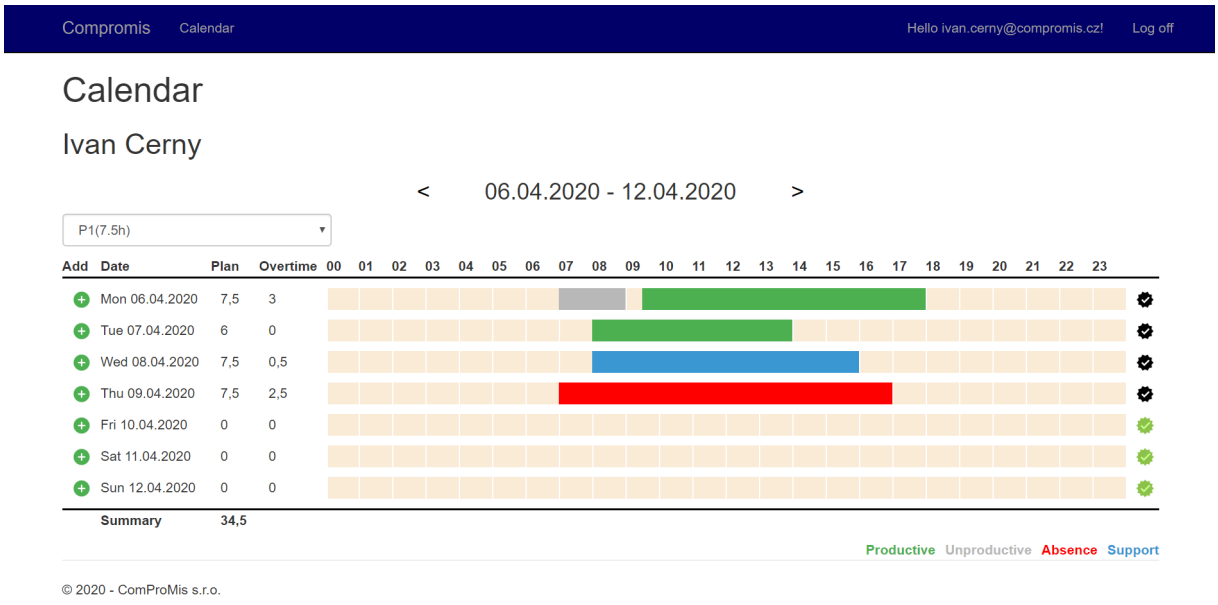
8.4 Aktivity

Spolu s prací na vizualizaci kalendáře, probíhala práce na přidávání aktivit, které se v kalendáři měly zobrazovat na časové ose. Pro přidání aktivit je použit formulář, který se zobrazí po kliknutí na daný pracovní den v podobě modálního okna. Formulář obsahuje pole pro výběr typu aktivity a pole pro zadání začátku a konce pracovní doby. Při přidávání aktivit bylo potřeba vyřešit také

pracovní podmínky vycházející ze smlouvy zaměstnanců, které říkají, že ranní příchody jsou zaokrouhlovány nahoru a odpolední odchody zaokrouhlovány dolů po 15 minutách. K tomuto účelu je zde použit jednoduchý přepočet, kdy se kontroluje, jestli zbytek minut po dělení 15 je nula. Pokud není, k zadaným minutám se připočítává, popřípadě odečítá potřebný počet minut. Při přidání nové aktivity probíhá kontrola, jestli nová aktivita nekoliduje s nějakou již přidanou aktivitou. V kladném případě se zobrazí uživateli upozornění a aktivita není přidána.

Kalendář a přidávání aktivit tvoří dohromady pohled na systém ze strany zaměstnance. Na obrázku 6 lze vidět finální verzi tohoto pohledu. Reálně je kalendář a časová osa tvořena HTML elementem <table>. Při zobrazování aktivit se přepočítá doba jejich trvání na minuty. Časová osa je pak rozdělena na buňky, kdy každá buňka reprezentuje 15 minut. Na základě toho, jestli se daný den v tuto dobu vyskytuje aktivita, se vykreslí buď prázdná buňka nebo buňka značící aktivitu. Barevné označení aktivit je pro rozlišení na čtyři základní skupiny aktivit. Legenda se nachází pod tabulkou. Pro zpřístupnění kalendáře je potřeba vybrat pracovní plán ze seznamu pod datem týdne. Na základě vybraného plánu se pak vypočítává každý den počet odpracovaných hodin spolu s hodinami nad rámec pracovního plánu. Aktivity jsou po přidání ve stavu neschválená. Stav aktivity značí dvojice ikon na konci řádku.

Při práci na zobrazování aktivit byl jejich vzhled zpočátku zvolen nevhodně. Prvotní nápad byl, aby každá aktivita měla svoje ohraničení. To výrazným způsobem zkomplikovalo celou logiku zobrazování. Tímto způsobem bylo potřeba kontrolovat, jestli se jedná o začátek aktivity, průběh aktivity nebo její konec. Každý stav měl jiný CSS styl. Proto byl nakonec zvolen jednodušší a zároveň grafický přijatelnější způsob bez ohraničení.

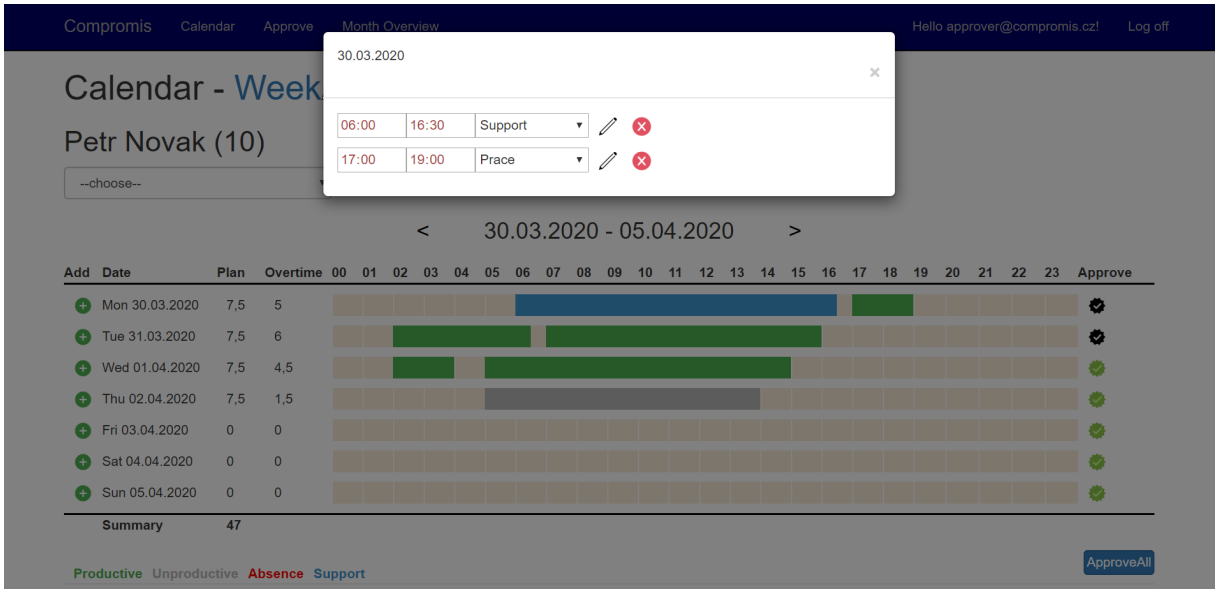


Obrázek 6: Pohled Zaměstnance

8.5 Schvalování

Schvalování aktivit zaměstnanců bylo dalším krokem při vývoji webové aplikace po dokončení pohledu ze strany zaměstnance. Při vývoji schvalovacího procesu byla nejdříve přidána funkce samotného schválení aktivity. V pozdější fázi obsahoval schvalovací proces i možnost editace aktivit zaměstnanců včetně odstraňování a přidávání aktivit, jak je vidět na obrázku 7. Schválení aktivit za daný den se provádí pomocí kliknutí na ikonu na konci řádku. Tmavá ikona značí neschválené aktivity, zelená schválené. Schválení všech aktivit za celý týden lze provést kliknutím na tlačítko „Approve All“. Schvalovací proces má dva režimy.

- Týdenní – Zobrazí se celý týden s aktivitami jednoho zaměstnance firmy. Tohoto zaměstnance je možné vybrat z menu.
- Denní – Zobrazí se jeden den a seznam všech zaměstnanců a jejich aktivit. Den lze vybrat taktéž z formuláře obsahující pole pro výběr data.



Obrázek 7: Schvalovací proces

8.6 Editace dat

Další práce na webové aplikaci se týkaly role datového manažera. Této osobě jsou přiděleny rozšířená práva pro editaci dat v databázi. Mezi jeho pravomoce patří například již zmíněná funkce pro generování dat v tabulce Kalendář. Dále editace pracovních plánů, typů aktivit, poboček firmy a zaměstnanců. Editací rozumíme možnost změny stávajících záznamů, odstranění záznamů a přidání nových záznamů. Tato část práce na projektu byla jedna z nejjednodušších. Funkce pro editaci jednotlivých tabulek byly založeny na velice podobném principu. Jednoduchost tohoto úkolu je zásluhou i použití Entity Frameworku.

Výpis 3 je ukázkou funkce pro přidání nového pracovního plánu, která se volá při odeslání formuláře. Pro přenos dat z formuláře se nejčastěji používá metoda **POST**. Funkce, která má být použita pro zpracování této metody, je označena atributem v hranatých závorkách. Jako parametr se této funkci předává objekt reprezentující formulář.

Pro vytvoření nového záznamu stačí vytvořit nový objekt typu požadované tabulky a nastavit jeho parametry údaji zadanými ve formuláři. Konkrétně u editace plánů bylo potřeba vyřešit problém rozdílného zápisu desetinného čísla. Pole pro zadávání desetinných čísel totiž vrací číslo ve formátu desetinné čárky. Nicméně tato hodnota nelze uložit do proměnné typu **double**. Proto se data z formuláře posílají ve tvaru textového řetězce a následně parsují na základě správného desetinného formátu. Poté se na instanci databázového kontextu a výběru z kolekce zavolá funkce **Add** s novým objektem. Nakonec se potvrzení změn provede pomocí funkce **SaveChanges**.

```
[HttpPost]
 public ActionResult AddPlan(PlanForm p)
 {
            if (ModelState.IsValid)
            {
                Plan plan = new Plan();
                plan.Id = Guid.NewGuid().ToString();
                plan.Kod = p.Kod;
                plan.Tydenni = p.Tydenni;
                plan.PocetHodin = double.Parse(p.PocetHodin,
                    System.Globalization.CultureInfo.InvariantCulture);
                plan.DobaOd = p.DobaOd;
                plan.DobaDo = p.DobaDo;

                _context.Plan.Add(plan);
                _context.SaveChanges();
            }
            else
            {
                TempData["error"] = "Incorrect format of given values.";
            }
            return RedirectToAction("Plan");
}
```

Výpis 3: Přidání nového plánu

Změna již vytvořených záznamů se provádí obdobně. První krok je získat editovaný objekt z databáze, upravit jeho parametry daty z formuláře a uložit změny. Odstranění objektu se provádí pomocí funkce **Remove** s předaným objektem.

8.7 Registrace a přihlašování

Registrace a přihlašování uživatelů bylo možná paradoxně provedeno až po vytvoření funkcí pro jednotlivé role zaměstnanců. Nicméně díky použití technologie Identity, která se stará o většinu operací spojených s uživatelskými účty, nebylo problémem nově vytvořeným účtům přidělit role. Vytvořeným funkcím pak stačilo akorát definovat, které role mají k dané funkci přístup. Ukázka přidělení funkce roli je ve výpisu 4.

```
[Authorize(Roles = "CanApproveActivity")]
 public ActionResult Schvalit(int? week)
 {
     if (User.IsInRole("CanApproveActivity"))
     {
     .
     .
     .
     }
}
```

Výpis 4: Ukázka přidělení role

Zavedení nového uživatele (Zaměstnance) do systému pak probíhá následovně. O zavedení do systému se stará datový manažer, který vyplňuje údaje o uživateli jako například jméno, příjmení, počet dnů dovolené... Mezi údaji dostane nový uživatel také přidělen firemní email s jednotným formátem. Registraci pak provádí samotný uživatel pomocí přiděleného emailu. Pokud nový uživatel zadá při registraci email, který existuje v systému a nemá přidělený žádný uživatelský účet, proběhne jeho registrace do systému. Jinak je upozorněn chybovou hláškou.

Komplikací při tomto postupu bylo ze začátku propojení tabulek Identity obsahující uživatelské účty a tabulek docházkového systému obsahující údaje o zaměstnancích. Tento problém se podařilo vyřešit rozšířením uživatelských účtů o atribut, který slouží jako odkaz na pracovníka z tabulky docházkového systému.

O zabezpečené přihlašování a správu uživatelských účtu se taktéž stará framework Identity, který automaticky registrovaným uživatelům po přihlášení vytváří možnost zobrazení uživatelského profilu. Profil nabízí standardní funkce jako je editace přihlašovacích údajů nebo například nastavení dvoufázového přihlašování přes mobilní telefon. K profilu byly přidány firemní informace zaměstnanců jako je zbývající počet dnů dovolené nebo počet stravenek na měsíc.

Registrace a přihlašování je aktuálně řešena pomocí osobních uživatelských účtů. Původně se mělo jednat o pracovní Azure účty, nicméně teď je projekt ve stavu, kdy bylo rozhodnuto o použití osobních účtů pro demonstrační účely.

8.8 Export dat

Součástí webové aplikace a jedním ze zadaných úkolů je taktéž rozhraní pro export dat do personálních a mzdových systémů. Pomocí tohoto rozhraní je možné zobrazit přehled odpracovaných hodin za jeden kalendářní měsíc. Tento přehled lze taktéž exportovat a stáhnout ve formátu .xsl.

Pro vytváření excelové tabulky jsem nejdříve chtěl použít knihovnu excellibrary, která je volně dostupná na internetu. Tato knihovna poskytuje intuitivní přístup při tvorbě excelových dokumentů. Nicméně po nějakém čase stráveném s touto knihovnou jsem narazil na její limity. Pomocí této knihovny lze jednoduše sázet data do buněk ve formě textu. Bohužel i přes to, že by knihovna měla obsahovat základní možnosti pro úpravu stylu, nebyl jsem schopen jakýmkoliv způsobem změnit vzhled buněk. Podpora ze strany vývojářů, podle repositáře na GitHubu, skončila někdy v roce 2016. Tudíž jsem pro lépe vypadají tabulku nakonec musel zvolit knihovnu Spire.XLS, která je taktéž dostupná v bezplatné verzi. S touto knihovnou již žádný problém ze strany požadovaných funkcí nebyl. Výsledná excelová tabulka je na obrázku 8.

Funkce pro export dat na základě zadání roku a měsíce zobrazí tabulku, která obsahuje seznam zaměstnanců spolu s daným kalendářním měsícem. Každý den obsahuje počet hodin chybějících do splnění základní pracovní doby, počet řádně odpracovaných hodin a počet hodin přesahující základní pracovní dobu. Na konci tabulky je součet všech těchto hodin. Tato tabulka by měla odpovídat tabulce, kterou sestavuje sekretářka na konci měsíce a na základě které pak vypočítává mzdu zaměstnanců. Funkce taktéž vypočítává povinný fond odpracovaných hodin za zvolený kalendářní měsíc. Tato funkce zároveň automaticky vytvoří excelový soubor a uloží jej na server. Při požadavku na stáhnutí tabulky ve formátu .xls se pouze vybere příslušný soubor a pomocí jednoduché funkce si může schvalovatel soubor uložit kamkoliv na svoje zařízení. Před zobrazením probíhá porovnání zvoleného měsíce a roku spolu s aktuálním datem. Pokud je zvolen měsíc větší než ten aktuální, je uživatel upozorněn hláškou. Výsledek funkce je na obrázku 9 (viz příloha A).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH
1	**Employee**	**Plan**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**Summary**
2		Under		7,5	7,5	7,5	7,5	7,5			7,5	7,5	7,5	7,5	7,5			7,5	7,5	7,5	7,5	7,5			7,5	7,5	7,5	7,5	7,5			7,5	7,5	165
3	Petr Novak	Normal																																0
4		Over																																0
5		Under		1,5	7,5	7,5	7,5	7,5			7,5	7,5	7,5	7,5	7,5			7,5	1,5	1,5	2,5	7,5			7,5	1,25	7,5	0,75	0,75					107,25
6	John Doe	Normal		6															6	6	5		4			6,25		6,5	6,5	7,5	7,5	7,5	7,5	76,25
7		Over																												0,5	1,25	0,75		2,5
8		Under		1,5	7,5	7,5	7,5	7,5			7,5	7,5	7,5	1,5	1,5			7,5	7,5	7,5	7,5	7,5			7,5	7,5	7,5	7,5	7,5			7,5	7,5	149,5
9	Carl Doe	Normal		6										6	6																			18
10		Over																																0
11		Under		7,5	7,5	7,5	7,5	7,5			1,5	7,5	7,5	1,5	7,5			7,5	7,5	7,5	7,5	7,5			7,5	7,5	7,5	7,5	7,5			7,5	7,5	153
12	Jan Novak	Normal									6			6																				12
13		Over																																0
14																																		
15																																		
16																																		

Obrázek 8: Excelový dokument

9 Závěr

9.1 Teoretické a praktické znalosti uplatněné v průběhu praxe

Rád bych vyzdvihl několik předmětů, které mi daly dobrý základ a znalosti při mém absolvování odborné praxe.

Při práci v ASP. NET MVC mi přišly vhod znalosti nabyté z předmětu Architektura Technologie .NET, jehož součástí byla právě i tvorba webových aplikací v nejnovější verzi ASP .NET, tedy verzi Core. Na praxi jsem sice nakonec zvolil vývoj aplikace ve verzi Framework, nicméně určité postupy byly aplikovatelné v obou těchto verzích.

Samozřejmě při tvorbě webové aplikace je potřeba znát základy HTML, CSS, Javascript. S těmito technologiemi jsem se mohl setkat v předmětu Vývoj internetových aplikací, který mi poskytl rozšíření mých znalostí v těchto technologiích.

9.2 Teoretické a praktické znalosti scházející v průběhu praxe

S technologií, kterou jsem se na praxi setkal úplně poprvé, byla Azure od Microsoftu, která měla sloužit, jak pro vývoj, tak pro finální nasazení aplikace. Při vývoji v teamu jsem se musel taktéž seznámit se službou Azure DevOps.

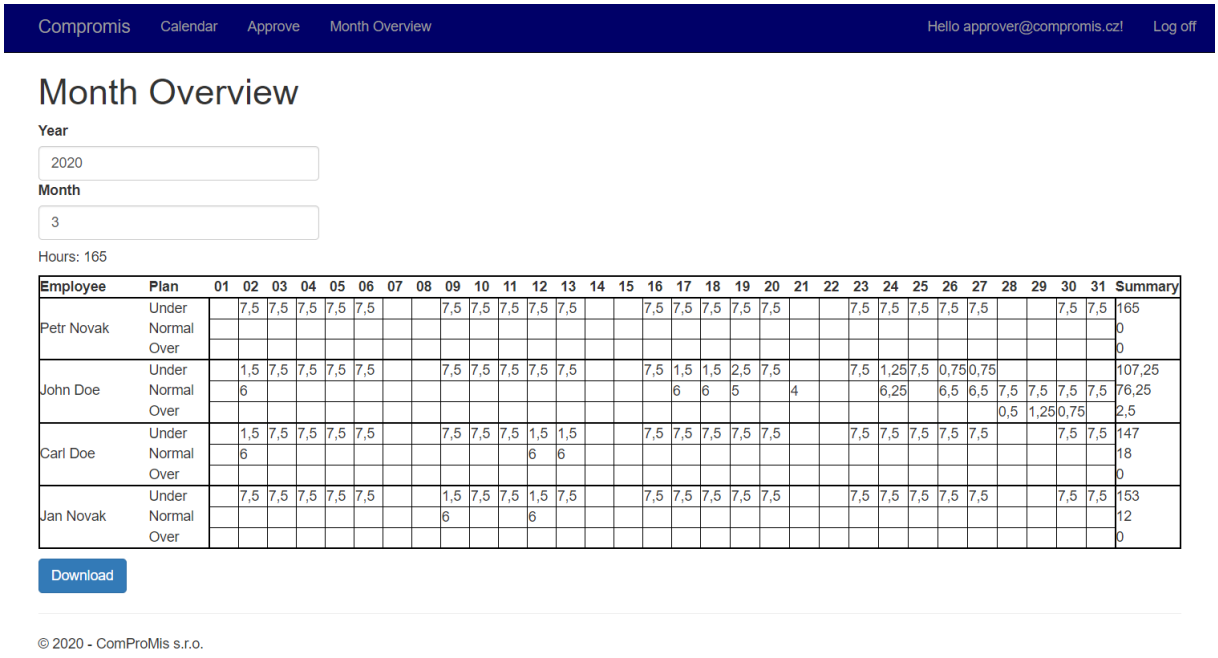
9.3 Dosažené výsledky a celkové zhodnocení

S výsledkem práce jsem celkově spokojen. Všechny dílčí úkoly, které projekt měl, se mi podařilo zvládnout i přes různé potíže, jako odchod kolegy nebo nutnost práce doma ke konci letního semestru. Oceňuji zejména praktické zkušenosti, které mi práce na tomto projektu přinesla. Jednak prohloubenou znalost ve vývoji webových aplikací, ale také schopnost prezentovat svoji práci před zaměstnavatelem. Součástí praxe byla taktéž práce v kolektivu, což taktéž hodnotím jako pozitivní zkušenost. Jsem rád, že projekt samotný mohl mít smysl i tím, že je to věc, která by se měla potencionálně začít používat ve firmě a nabídnout zaměstnancům firmy modernější a efektivnější způsob vedení a evidenci docházky.

Literatura

1. *Compromis, O nás* [online] [cit. 2019-04-20]. Dostupné z: https://www.compromis.cz/new.
2. *Microsoft Azure* [online] [cit. 2019-04-25]. Dostupné z: https://azure.microsoft.com/cs-cz/.
3. *Microsoft Azure* [online] [cit. 2019-04-20]. Dostupné z: https://cs.wikipedia.org/wiki/Microsoft_Azure.
4. *SQL Azure Create Database Tutorial* [online] [cit. 2019-04-20]. Dostupné z: https://www.mssqltips.com/sqlservertip/5172/sql-azure-create-database-tutorial/.
5. *MS SQL Server* [online] [cit. 2019-04-25]. Dostupné z: https://www.microsoft.com/cs-cz/sql-server/.
6. *MS SQL databáze* [online] [cit. 2019-04-20]. Dostupné z: https://napoveda.czechia.com/clanek/ms-sql-databaze/.
7. *ASP.NET MVC Pattern* [online] [cit. 2019-04-25]. Dostupné z: https://dotnet.microsoft.com/apps/aspnet/mvc.
8. *ASP.NET MVC Framework* [online] [cit. 2019-04-20]. Dostupné z: https://cs.wikipedia.org/wiki/ASP.NET_MVC_Framework.
9. *Razor syntax reference for ASP.NET Core* [online] [cit. 2019-04-20]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/mvc/views/razor?view=aspnetcore-3.1.
10. *ASP.NET Identity* [online] [cit. 2019-04-20]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/identity/.
11. *Entity Framework documentation* [online] [cit. 2019-04-25]. Dostupné z: https://docs.microsoft.com/cs-cz/ef/.
12. *Entity Framework* [online] [cit. 2019-04-25]. Dostupné z: https://en.wikipedia.org/wiki/Entity_Framework.
13. *Dokumentace k jazyku C #* [online] [cit. 2019-04-25]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/csharp/.
14. *C Sharp* [online] [cit. 2019-04-20]. Dostupné z: https://cs.wikipedia.org/wiki/C_Sharp.

A Rozhraní pro export dat



Employee	Plan	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Summary
Petr Novak	Under		7,5	7,5	7,5	7,5	7,5			7,5	7,5	7,5	7,5	7,5			7,5	7,5	7,5	7,5	7,5			7,5	7,5	7,5	7,5	7,5			7,5	7,5	165
	Normal																																0
	Over																																0
John Doe	Under		1,5	7,5	7,5	7,5	7,5			7,5	7,5	7,5	7,5	7,5			7,5	1,5	1,5	2,5	7,5			7,5	1,25	7,5	0,75	0,75					107,25
	Normal		6															6	6	5		4			6,25		6,5	6,5	7,5	7,5	7,5	7,5	76,25
	Over																												0,5	1,25	0,75		2,5
Carl Doe	Under		1,5	7,5	7,5	7,5	7,5			7,5	7,5	7,5	1,5	1,5			7,5	7,5	7,5	7,5	7,5			7,5	7,5	7,5	7,5	7,5			7,5	7,5	147
	Normal		6										6	6																			18
	Over																																0
Jan Novak	Under		7,5	7,5	7,5	7,5	7,5			1,5	7,5	7,5	1,5	7,5			7,5	7,5	7,5	7,5	7,5			7,5	7,5	7,5	7,5	7,5			7,5	7,5	153
	Normal									6			6																				12
	Over																																0

Obrázek 9: Měsíční přehled

B Zdrojové kódy

```
[HttpPost]
public ActionResult Generate(KalendarForm form)
{
    if (ModelState.IsValid)
    {
        Thread.CurrentThread.CurrentCulture = new CultureInfo("cs-CZ");
        DateTimeFormatInfo dfi = DateTimeFormatInfo.CurrentInfo;
        Calendar cal = dfi.Calendar;

        DateTime start = form.StartDate;
        DateTime end = form.EndDate;

        while (start <= end)
        {
            Kalendar den = new Kalendar();
            den.Id = Guid.NewGuid().ToString();
            den.Datum = start;
            den.DenVTydnu = (int)start.DayOfWeek == 0 ? 7 : (int)start.
                DayOfWeek;
            den.StatniSvatek = DateSystem.IsPublicHoliday(den.Datum,
                CountryCode.CZ) ? true : false;

            den.KalendarniTyden = cal.GetWeekOfYear(start, dfi.
                CalendarWeekRule, dfi.FirstDayOfWeek);

            DateTime lastadded = _context.Kalendar.Max(item => item.Datum
                );
            if (lastadded >= den.Datum)
            {
             TempData["error"] = "Day in the range is already inserted.";
             return View("GenerateCalendar", form);
            }
            try
            {
                _context.Kalendar.Add(den);
                _context.SaveChanges();
            }
```

```
                catch (System.Data.Entity.Infrastructure.DbUpdateException e)
                {
                    Console.WriteLine(e.Message);
                }

                start = start.AddDays(1);
            }
        }
        else
        {
            return View("GenerateCalendar",form);
        }
        return RedirectToAction("Index","Kalendar");
    }
```

Výpis 5: Generování záznamů tabulky Kalendář